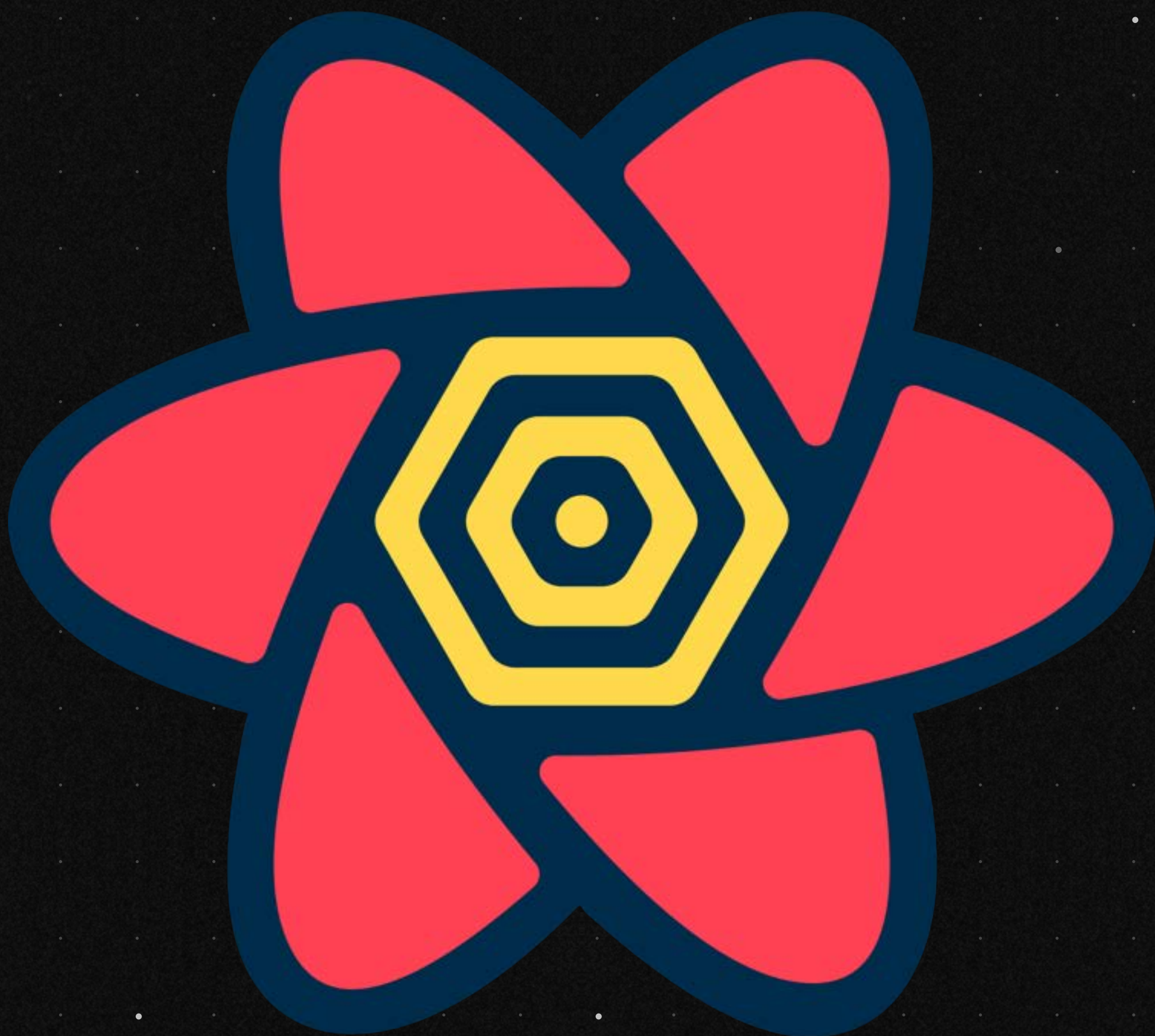
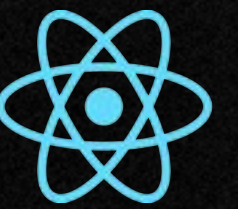


# React Query







**React Query is a powerful library that helps you manage and fetch data in React applications.**

## Installation

To get started with React Query, you need to install it in your project. You can use npm or yarn to install the package.

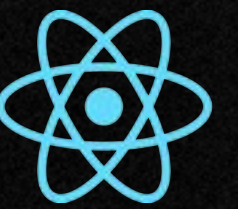


```
npm install react-query
```



Swipe →





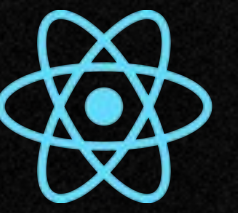
# Importing and setting up React Query

```
import { QueryClient, QueryClientProvider,
useQuery } from 'react-query';
import { ReactQueryDevtools } from 'react-
query/devtools';

const queryClient = new QueryClient();

function App() {
  return (
    <QueryClientProvider client={queryClient}>
      <ExampleComponent />
      <ReactQueryDevtools />
    </QueryClientProvider>
  );
}
```





# Fetching data using useQuery hook

The useQuery hook is the primary way to fetch data in React Query.

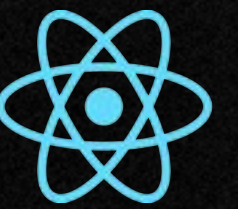


```
function ExampleComponent() {  
  const { isLoading, error, data } =  
  useQuery('todos', () =>  
    fetch('json-link').then((response) =>  
      response.json()  
    )  
  );  
}
```

It takes a query key and an async function (or a promise) that fetches the data.







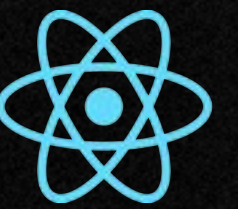
## Handling loading and errors-



```
if (isLoading) {  
  return <div>Loading...</div>;  
}  
  
if (error) {  
  return <div>Error: {error.message}</div>;  
}  
  
return (  
  <div>  
    {data.map((todo) => (  
      <div key={todo.id}>{todo.title}</div>  
    ))}  
  </div>  
);  
}
```







# Caching data:

React Query automatically caches the fetched data, which improves performance by avoiding unnecessary API calls.

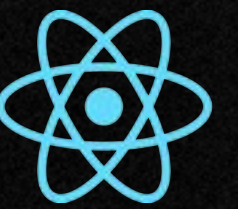


```
function ExampleComponent() {  
  const { isLoading, error, data } =  
  useQuery('todos', () =>  
    fetch('json-link').then((response) =>  
      response.json()  
    )  
  );  
  
  // Rest of the code...  
}
```

If the same query key is used in multiple components, React Query will reuse the cached data!







# Invalidating and refetching data

React Query provides a way to manually invalidate and refetch data.



```
import { useQueryClient } from 'react-query';

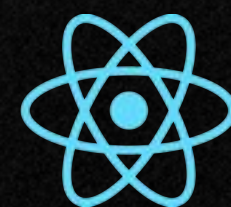
function ExampleComponent() {
  const queryClient = useQueryClient();

  const handleLogout = async () => {
    // Invalidate and refetch all queries with
    'todos' query key
    await queryClient.invalidateQueries('todos');
    await queryClient.refetchQueries('todos');
  };

  // Rest of the code...
}
```









To better understand this topic  
check out [this video](#) on my **YouTube**  
channel.





**React Query Tutorial | #34**  
**React Course**  
1K views • 11 days ago  
Code with Sloba  
In this tutorial, you will learn how to make API reque...  
Intro | React... 10 chapters



**Code with Sloba**  
@CodewithSloba  
13.2K subscribers

Subscribe



Swipe →





codewith**sloba**.com

Get a weekly digest of my tips and  
tutorials by subscribing now.